1. RDF Model and SPARQL RDF Terms Syntax

RDF Graph:
A set of RDF Triples

RDF Triple:
A triple (3-tuple) of:

Subject:
URI or Blank Node

Predicate:
URI

Object:
URI or Blank Node or Literal

URI:
An absolute URI which may include a # fragment.

<http://www.w3.org/>
<http://example.org/#fragment>

Relative URI resolved against base URI.

<> Base URI, usually the query document URI.

RDF Literal:
A Unicode string with an optional language tag.

"hello"
"bonjour"@fr

RDF Typed Literal:
A Unicode string and datatype URI for encoding datatypes.

"abc"^^<http://example.org/myDatatype>
abbreviated with an XML QName style as:

"10"^^xsd:integer
1.2e3
"true"^^xsd:boolean

Blank Node:
A node in a graph with a local name. The scope of the name is the RDF graph.

_:node

2. Popular RDF Namespaces

Namespace Common Prefix Namespace URI
RDF rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
Dublin Core dc: http://purl.org/dc/elements/1.1/
FOAF foaf: http://xmlns.com/foaf/0.1/
XML Schema xsd: http://www.w3.org/2001/XMLSchema#
RDFS rdfs: http://www.w3.org/2000/01/rdf-schema#
OWL owl: http://www.w3.org/2002/07/owl#

3. SPARQL Language Reference


RDF Term:
A part of an RDF Triple. A URI, Blank Node or a Literal.

<uri> _:bl "Literal"@en "abc123"^^my:datatype

Query Variable:
Identifiers for binding to RDF Terms in matches.

?a / $b or in lists: $name $title $place

Triple Pattern:
An RDF Triple with Query Variables allowed in each term:

<http://example.org/abc> ?x "Hello"

Turtle abbreviations can be used for Triple Patterns, see Section 4.

Graph Pattern:
A block that matches part of the queried RDF graph.

Basic
A set of Triple Patterns which binds RDF Terms in the graph to variables.

Graph Pattern:
Written as a (...) block with ',', separating the triple patterns:

{ <http://example.org/abc> ?y "Hello" .
  ?subject $predicate "Literal" }

Group
Graph Pattern:
A graph pattern that contains multiple graph patterns which must all match
to provide a result.

{ { ?person rdfs:type foaf:Person }
  { ?person foaf:name "Dave" } }

Optional
Graph Pattern:
A graph pattern which may fail to match and provide bindings but not
cause the entire query to fail. Written with the OPTIONAL keyword before a graph pattern.

OPTIONAL { ?person foaf:nick ?nick }

Union
Graph Pattern:
A pair of graph patterns any of which may match and bind the same
variables. Written with the UNION keyword between two graph patterns.

{ ?node ex:name ?name } UNION
{ ?node vcard:FN ?name }

Graph
Graph Pattern:
A keyword for specifying a graph name to use or to return a graph name
as a binding. Written with the GRAPH keyword before a graph pattern.

GRAPH <http://example.org/myfoaf>
{ { ?person foaf:name ?name } GRAPH ?graph { ?person foaf:name ?name } }

Value Constraints:
A boolean expression in a graph pattern over query variables that
constrains matched graph patterns.

{ ?item ex:size $size . FILTER $size < 10 }
5. SPARQL Query Result Forms

Variable Bindings: A sequence of (set of variable bindings) for each query pattern match.

SELECT *
WHERE {?a rdf:type ?b}
to ask for bindings for all variables mentioned in the query and
SELECT ?a ?b
WHERE {?a rdf:type ?b}
to list them explicitly.

RDF Graph: An RDF graph describing resources either given by URI
Describe
Resources: DESCRIBE <http://example.org/thing>
or by binding variables using the same syntax as SELECT.
DESCRIBE ?person
WHERE {?person foaf:name "Dave"}
Build an RDF graph made by substituting variables into a triple template.
CONSTRUCT {?a foaf:knows ?b}
WHERE {?a ex:KnowsQuiteWell ?b}
Boolean: True if the query pattern could be answered.
ASK
WHERE {?a rdf:type foaf:Person}

6. Query Results Controls and Sorting

The optional controls on query results are optionally performed in the following order:
1. DISTINCT to ensure solutions in the sequence are unique
2. ORDER BY ordering solutions sequences by variable or function call:
   ORDER BY DESC(?date) ?title ASC(?familyName)
in descending order by date, by title, by familyName ascending
3. LIMIT n to restrict the number of solutions to n
4. OFFSET m to start the results in the solution from item m

7. Values – datatypes, expressions and operators

Supported datatypes: RDF Terms, xsd:boolean, xsd:string, xsd:double, xsd:float,
xsd:decimal, xsd:integer and xsd:dateTime

Logical operators: Logical: A | B, A & B, ! A
Comparison (A op B): =, !=, <, >, <=, >=
Arithmetic operators: Unary: +A, -A
Binary (A op B): *, /, +,
Comparison:
RDF operators: Boolean: BOUND(A), ISURI(A), ISBLANK(A), ISLITERAL(A)
String: STR(A), LANG(A), DATATYPE(A)
String Match operator: REGEX(string, pattern [ flags])
Extension Functions and Explicit Type Casting:
Automatic Type: from xsd:decimal to xsd:float
Promotion: from xsd:float to xsd:double

8. Turtle RDF Syntax Reference

Turtle (Terse RDF Triple Language) describes triples in an RDF graph and allows abbreviations. Triple Patterns in SPARQL can use the same abbreviations.

This description is based on Turtle 2004-12-23 from <http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/>

RDF Terms: URI: <URI>
Literal: "string" or "string"@language or ^^<datatype URI>
Blank Node: _: name or [] for an anonymous blank node

@prefix operator: URIs can be written as XML-style QNames by defining a prefix / URI binding:
@prefix dc: <http://purl.org/dc/elements/1.1/> .

Triples: Written as 3 RDF terms with whitespace separating them as necessary, and ',,' between triples:
<> dc:title "SPARQL Reference".

; operator: Triples with the same subject and predicate may be abbreviated with ',':
<> dc:title "My Book", "Mein Buch"@de .

; operator: Triples with the same subject may be abbreviated with ';':
<> dc:title "My Workplace";
   dc:publisher "My Employer" .

[ ... ] operator: A sequence of (predicate object) pairs may be put inside [ ... ] and a blank node subject will be assigned to them:

[] operator: A blank node:

a predicate: The often-used rdf:type QName may be abbreviated by the keyword a as a predicate:
<> a Foaf:Document .

Integers: Decimal integers 0 or larger can be written directly as literals (type xsd:integer):
<> ex:sizeInBytes 12345.

(... ) collections: RDF collections can be written inside ( ... ) as a space-separated list of the contents:

9. Example SPARQL Query

BASE <http://example.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
# This is a relative URI to BASE above
PREFIX ex: <http://example.org/mybook/>

SELECT DISTINCT ?person ?name ?age
FROM <http://rdf.example.org/people.rdf>
WHERE {?person a foaf:Person ;
          foaf:name ?name .
          OPTIONAL { ?person ex:age ?age } .
          FILTER ! REGEX(?name, "Bob")
        }
LIMIT 3